A Quick Reference for The Debugger V2 - 2/21/89 © Steve Jasik 1989

Installation Procedure

1) Copy the **WHITE** Floppy disk to your Hard disk (and make a backup of the floppy).
   The default ROM snt file setup is for a Mac+ on the "+/SE" and a Mac II on the Univ Disk.
   If your configuration is other than that, then copy the correct ROM snt file into the "Dbgr/Nosy files" folder from the "Mac SE files" or from the blue disk (Univ Version).
2) Copy the files in **"put files in System Folder"** to your System Folder.
3) Run **Nosy**, and create ".snt" files for the **Finder** and **MultiFinder.**
4) If you want to **debug MPW Tools**, then create an ".snt" file for the "MPW Shell" , copy the file "InformDbgr" to to Tools folder in MPW, and add a line to "UserStartup" to execute it.
5) Adjust the names of INITs as per the next section.
6) Boot your machine and let it rip.
7) Optionally you may customize "ROM/CODE.snt" to your system configuration by disassembling "ROM" (in the Nosy/Dbgr files folder) and saving the .snt file.
   If you have any problems with the installation, then see pages 8 - 9 of the manual.

MultiScreen Operation

Mac II, IIx - Debugger takes over the startup Screen. Option-Monitors in Control Panel to set.
Mac +/SE - Only with E-Machines Monitor, install Big Picture™ INIT from Dbgr disk.

RunDbgr versus XDbgr_Startup

You need at least 2 Meg of RAM to run *The Debugger* in the background and 4 Meg or more is preferred to get any real work done. You can **NOT** Launch *The Debugger* after MultiFinder. Mac SE's with third party accelerator cards (Radius, MacPeak): you may have to patch the System to get The Debugger to launch at INIT time. Read the file "Radius Notes" for details.

Macsbug, INITs and Other Gotchas

The Macsbug in "put files in System folder" installs itself in high memory and records patches to the Traps until the Finder is launched. It takes up 8K of memory. To see who patched the Traps, check out the "ROM Patch Info" command in the Display menu. Diehard fans of Apple's Macsbug may rename it "Disassembler" to get it installed.

When *The Debugger* starts up it makes a copy of the "world" (trap vectors, etc.) so that it **will not be affected by patches to the traps** that are set by programs that come after it. To debug INITs that come alphabetically before *xDbgr_Startup* you may rename it or the INIT.
The Debugger is not compatible with all INITs. In particular "SuitcaseII" should be renamed to "zSuitcaseII" so that it loads after *The Debugger*. An alternative is to use Master Juggler which is compatible with *The Debugger* ; it can be used to invoke Desk Accessories inside it. **Note that you use DAs inside *The Debugger* at your own risk!!!**
To avoid interference between the command-option key equivalents in QuicKeys™ and those in *The Debugger*, you may wish to rename QuicKeys™ to zQuicKeys™ .

Menus and Commands, General

Command-option key equivalents show up as **Omega**-Key (Ω) in the Menus.
When the key equivalent is a **lower case letter**, Command-shift-letter will result in a different function (Cmd-B is a simple Bkpt, Cmd-shift-B is a Bkpt with the current action clause attached to it.)
Some unshifted/shifted menu items options are presented in the menu in the format "unshifted/SHIFTED command." For example, in the *Tables* menu, the first menu item is **Record/ALL names**, which lists all the known type names if the shift key is held down, and only the built in Record names if is not.
Menu items that are toggles show up in **Boid face**; a check mark indicates it is selected.

Many commands check for a selection in the front window, and will use it if present.
If nothing is selected, then the command will bring up a simple type in dialog box just below the menu bar. You can **paste into these dialogs** with Cmd-V; and dismiss them with a Cmd-Q. Clicking in the **"Cmd:"** box is equivalent to a return.
Use **Cmd-period** to abort a command or stop continuous Step mode, etc.
The first item in the menu bar is the amount of free space in *The Debugger*'s Heap zone.
The first menu contains a **Help** command.

Windows & Tasks, General

All windows (not dialogs) are Text Edit based (no tabs, 64K size limit). The normal rules for selection (double-click, shift-click, etc.) hold for them. Undo is not implemented.
Use **Option-Click** to obtain a popup menu of the windows to bring one to the front.
In *The Debugger* most of the windows are read only to protect the data from inadvertent modification. You can copy from them.
A quick way of transferring a selection to the "-Notes-" window is to use **Cmd-N.**
Windows that are **task relative** are prefixed with "∂taskname."
A **task** is a process (APPL or executable resource). The list of tasks known to *The Debugger* is displayed in the **"-Task & File Info-"** window (Ω-T) along with the list of open files.
The last item in the Menu bar is the number and name of the task you are in. It will change as you switch from one task relative window to another. Commands that are task relative (such as display Code | Data block) will search tables, etc. relative to that task.

To debug a task **symbolically** you must prepare an **auxiliary file** that contains symbol information that resides in the **same folder from which the task will be executed.**
The aux file may be: a Nosy "snt" file, an MPW or MDS style ".map" file, an MPW V3 ".SYM" file or the LightspeedC™ PROJ file itself.

The Find command - Searching for Text in a Window

If some object is selected in the front window, then the Find command (Cmd-F) accepts it as the search pattern and looks for the next occurrence of it as a token (separated by delimiters).
If nothing is hilited in the front window, then it brings up the find dialog window to give you more flexibility.

Debugger, Entry/Exit

Cmd-E to exit *The Debugger* to the Problem Program (PP).
To transfer to control from the PP to *The Debugger*, press **Option-** or the Programmers Switch, or use the Programmer's Key INIT.
When in *The Debugger*, use the ~ (tilde) key to view the PP's screen (single screen systems).

On Entry to The Debugger from the PP

On swapping into its world *The Debugger* puts one or more lines into the **"-Notes-"** window to show why you entered it, updates the **"-Registers-"** and **"-Stack State-"** windows, and a window of the procedure containing a disassembly (or source) of the current PC.
The menubar task indicator is set to the task of the procedure containing the PC.
Note: To reset the display of the proc to the current PC, hilite the value of the PC in the **"-Registers-"** window and Cmd-D.

Source Level Debugging (MPW V3 .SYM file or LightspeedC™ project)

For a LightspeedC project, the **source files must be on the same volume** as the "project" file, while with MPW ".SYM" files, the source is assumed to be from where it was built.
Use the "Source only windows" command (Ω-V) to set the default mode of display.
Use a **Command-click** (mouse down) to toggle a window between Source only and source with interspersed assembly. In the source only windows, a"Σ" in column 1 denotes the beginning of a statement. In LightSpeedC **Compile with Debug on** so *The Debugger* has the source line/code info, etc. Running with the LightSpeedC Debugger is optional.

Displaying procedures

Hilite the name of a procedure and **Cmd-D**, or Cmd-D with nothing hilited and you will get a type in dialog box. More frequently in *The Debugger*, you want to look at an active procedure, i.e. one that is in the call chain. Activate the **"-Stack State-"** window, find the procedure you want, hilite the "=nnn" and Cmd-D. Not only is this slightly faster, but it also switches *The Debugger* to the task the procedure is in and positions the display to the point of the call.

Modifying Memory or one of the Registers

Use the Set... (**Cmd-Y**) command and type in Rxi=hexval OR address=hexval/size
where size is **Byte**, **Word** or **Long.**

Breakpoints

To set or clear a breakpoint, hilite a line in a procedure window and **Cmd-B**.
To attach the current "action clause" to a breakpoint, **Cmd-shift-B**.
To define an "action clause", hilite it and **Ω-B**. It will become the current "action clause".
To change the current "action clause", de-hilite any selection in the front window and **Ω-B**.
Then enter the number of the "action clause" you wish to make current.
To delete an "action clause". de-hilite any selection in the front window and **Ω-shift-B.**
Then enter the number of the "action clause" you wish to delete.
Do not set a breakpoint in ROM patch code that The Debugger may execute!

Controlled execution (stepping)

You may **single step by Cmd-comma** (Step Through) or Cmd-? (Step Into) or you may step continuous by Cmd-semicolon (Step Continuous). To **bypass the Step continuous dialog**, use **Cmd-shift-semicolon**. When in continuous step mode, hold down the Cmd key to **pause** execution and Cmd-period to stop it. If you step into a procedure that you don't want to be in, use Cmd-quote (Go Until Caller) or **Ω-R** (Go Until Exit ROM) to get out of it.
To paraphrase Charles Dickens: Tis a far, far better thing that you step continuous, then trying to single step and lift your fingers off the Cmd and comma or "?" keys every time.

Changing the Program Counter (PC)

To change the Program Counter (PC), hilite a line containing the new PC, and use the Set PC (**Cmd-P**) or Go Until PC (**Ω-P**) command. You do this in **Source only windows at your own risk**, as the compilers may have generated code that initializes registers in non obvious places.

Displaying memory, values of variables, Hypertext

Hilite an address in any window and **Cmd-space** to obtain a Hex/Ascii display of memory.
To display memory according to the Type "TTT", hilite an expression of the form:
"TTT@nnn" and Cmd-space. nnn may be any expression.
Use **Cmd-8** to toggle a Cmd-space window between one that is static and one that is updated on each entry to *The Debugger*.
To display the **definition of a Type**, Cmd-space and enter the Typename.
To display the **value of a global variable**, switch to the task containing the global, hilite the name, and **Cmd-W**. The name, its value, etc. will appear in the **"-Values-"** window.
To **typecast** a name, **Cmd-W** and enter "name:Type"
To **delete a name** from the values window, hilite the name and **Cmd-shift-W**.
To view the values of the **Local variables** of the active procedures, **Cmd-shift-U**.

Watching memory for changes

First determine the hexadecimal address of the variable that you want to have watched from an "asm" window or placing the variable in the values display, then **Ω-W**, and enter the address of the variable and the number of bytes that you want to watch. On pressing return, *The Debugger* will exit to the problem program. The manual describes other variations that are possible, such as watching a section of memory within a heap block, etc.

Intercepting Traps

You may set a trap intercept by hiliting the name of a trap and **Cmd-J**, or Cmd-J and type its name, or **Cmd-shift-J** for a fancy dialog to select trap names by suite or name.
To **clear an intercept**, Hilite the trap name or line containing it and **Ω-J**.

Trap Discipline and Handle Zapping - the Bondage Menu

Trap discipline checks the arguments to a mac trap call on entry for consistency, and breaks into *The Debugger* with a **message in the "-Notes-" window** if they are incorrect.
Cmd-5 (Ignore Current Error) will suppress future Discipline messages for the Trap call at PC.
Handle Zapping is really a sub menu item of discipline. When it is selected, blocks in the heap will be set to a value that should cause an address error if the block is referenced after it is released (_DisposPtr, _DisposeHandle). See the manual for more details.

Heap Display & the Where the F... is it Command

Cmd-H to bring up a display of the Heap zone of the current task. **Cmd-shift-H** to display the System Heap zone. To find out what heap zone and block an address is in, hilite it and **Ω-G**.

The Expression Language

A language based on the Pascal Syntax, it contains arithmetic operators, references to registers (Rxi), user variables (name or ∂taskname.Varname), constants (default is hex, #nn for decimal base), debugger variables and functions (prefixed with a "?"), and for the curious, who insist on knowing where their procedures are in memory, the expression "@procname" will return the address of loaded procedure.
For a full description and examples consult the Help file or the manual.

".dsi" Files and Debugger Flags

".dsi" files are documented in the file Tutorial.dsi and the Manual, check it out.
Debugger flags are described in the "ROM.dsi" file. **Ω-Z** to see their current values.
To **change a flag value**, select lines of the form "=Flag" , "flag_name = 0 or 1;" and **Ω-F**.

Debugging MPW Tools

Is automatic if you have followed step 4 of the installation procedure and added a line to your UserStartUp" file to execute "InformDbgr" once per Launch of the MPW Shell.

Debugging INITs

You debug an INIT's setup code by supplying an auxiliary file. To symbolically debug the Traps that the INIT patches, rebuild the ROM "snt" file with the INIT installed.

Debugging Programs that jump off into the wild blue yonder

If you have a computer with an 020/030 CPU then turn on **"Trace Jumps"** in the **Go"** menu and rerun the program. If you don't then go out and borrow, buy or steal one.

Debugging Programs that blow off in the ROM

Step back from the computer, take a deep breath and count to ten. Find a Voodoo doll that looks like an Apple ROM programmer and stick some pins into it. If it is past 1 AM then go directly to bed, do not pass GO. More seriously if the problem is repeatable, as most bugs are, then rerun the program with Discipline on, and if you have a computer with an 020/030 CPU then turn on **"Trace Jumps"** in the **Go"** menu. If you are not me, then avoid trying to step into the ROM code as you will just get lost in the Rats nest that it has become. Look at the code in your program, read the accumulated tech notes, Inside Macintosh Volumes 1 through 5, etc until you can pin the blame upon your program. All else failing, call Mac DTS or John-Louis Gasse and tell them what you think of their @#$!& system.

Terminating the current debugging session

When the sushi has hit the fan or the magic Bomb box has appeared, you should consider the "Boot System" or "Flush System and Boot" commands as viable alternatives to continuing the current masochism. If you don't want the Volumes flushed on a Mac II/IIx, then use the Reset switch on the side of the Machine.

Terminating an Application (Multifinder mode only)

To do this the PC must be within your APPLications heap zone, so adjust it first if necessary, then select the Shutdown APPL command from the first menu.

Shutting down the Debugger (RunDbgr mode only)

This operation can be done with relative impunity in almost all cases.
Use the "Shutdown Debugger " command from the Files menu to terminate your application and *The Debugger*, and transfer to the Finder, MPW or LightSpeedC.

A last Word - don't say I didn't tell ya so

Make a copy of your System file on your hard disk, and keep a Bootable floppy disk with some recovery utilities handy. MPW's *Rezdet* will detect corrupted System files, etc.

Controlled execution (stepping)

You may **single step by Cmd-comma** (Step Through) or Cmd-? (Step Into) or you may step continuous by Cmd-semicolon (Step Continuous).
To **bypass the Step continuous dialog, use shift-Cmd-semicolon.** When in continuous step mode, hold down the Cmd key to **pause** execution and Cmd-period to stop it. If you step into a procedure that you don't want to be in, use Cmd-quote (Go Until Caller) or Ω-R (Go Until Exit ROM) to get out of it.
To paraphrase Charles Dickens: Tis a far, far better thing that you step continuous, then trying to single step and lift your fingers off the Cmd and comma or "?" keys every time.

Changing the Program Counter (PC)

To change the PC, hilite a line containing the new PC, and use the Set PC (**Cmd-P**) or Go Until PC (**Ω-P**) command. You do this in **Source only windows at your own risk,**as the compilers may have generated code to initialize registers in non obvious places.

Modifying Memory or one of the Registers

Use the Set... (**Cmd-Y**) command and type in 'Rxi=hexval' OR 'address=hexval/size'
where size is **Byte, Word** or **Long.** The syntax 'address:hexval' is also accepted.

Displaying memory, values of variables, Hypertext

Hilite an address in any window and **Cmd-space** to obtain a Hex/Ascii display of memory.
To display memory according to the Type "TTT", hilite an expression of the form:
"TTT@nnn" and Cmd-space. nnn may be any expression.

Use **Cmd-8** to toggle a Cmd-space window between one that is static and one that is updated on each entry to *The Debugger*.

To display the **definition of a Type,** Cmd-space and enter the Typename.

To display the **value of a global variable,** switch to the task containing the global, hilite the name, and **Cmd-W.** The name, its value, etc. will appear in the "**-Values-**" window.

To **typecast** a name, **Cmd-W** and enter "name:Type"
To **delete a name** from the values window, hilite the name and **Cmd-shift-W.**

To view the values of the **Local variables** of the active procedures, **Cmd-shift-U.**

Watching memory for changes

First determine the hexadecimal address of the variable that you want to have watched from an "asm" window or by placing the variable in the values display, then **Ω-W,** and enter the address of the variable and the number of bytes that you want to watch. On pressing return, *The Debugger* will exit to the problem program. The manual describes other variations that are possible, such as watching a section of memory within a heap block, etc.

Intercepting Traps

You may set a trap intercept by hiliting the name of a trap and **Cmd-J,** or Cmd-J and type its name, or **Cmd-shift-J** for a fancy dialog to select trap names by suite or name.

To **clear an intercept,** Hilite the trap name or line containing it and **Ω-J.**

Trap Discipline and Handle Zapping - the Bondage Menu

Trap discipline checks the arguments to a mac trap call on entry for consistency, and breaks into *The Debugger* with a **message in the "-Notes-" window** if they are incorrect.

Cmd-5 (Ignore Current Error) will suppress future Discipline messages for the Trap call at PC.
When **Handle Zapping** is selected, blocks in the heap will be set to a value that should cause an address error if the block is referenced after it is released (_DisposPtr, _DisposeHandle). See the manual for more details.

Heap Display & the Where the F... is it Command

Cmd-H to bring up a display of the Heap zone of the current task.
shift-Cmd-H to display the System Heap zone.
To find out what heap zone and block an address is in, hilite the address and **Ω-G.**

The Expression Language

A language based on the Pascal Syntax, it contains arithmetic operators, references to registers (Rxi), user variables (name or ∂taskname.Varname), constants (default is hex, #nn for decimal base), debugger variables and functions (prefixed with a "?"), and for the curious, who insist on knowing where their procedures are in memory, the expression "@procname" will return the address of loaded procedure.
For a full description and examples consult the Help file or the manual (pages 37 - 40).

".dsi" Files and Debugger Flags

".dsi" files are documented in the Manual (pages 33 - 34) and DTN #3, check them out.
Debugger flags are described in the "ROM.dsi" file. **Ω-Z** to see their current values.
To **toggle a flag value,** select a flag_name, hilight it and **Ω-F.**

Debugging MPW Tools

Is automatic if you have followed step 8 of the installation procedure (DTN #0) and added a line to your UserStartUp" file to execute "InformDbgr" once per Launch of the MPW Shell.

Debugging INITs

You debug an INIT's setup code by supplying an auxiliary file. Read DTN #5 for more info.

MMU Protection

Is descirbed in DTN #7, read it.

Debugging Programs that jump off into the wild blue yonder

If you have a computer with an 020/030 CPU then turn on **"Trace Jumps"** in the **Go"** menu and rerun the program. If you don't then go out and borrow, buy or steal one. Jump Tracing does NOT work on 040's.

Debugging Programs that blow off in the ROM

Step back from the computer, take a deep breath and count to ten. Find a Voodoo doll that looks like an Apple ROM programmer and stick some pins into it. If it is past 1 AM then go directly to bed, do not pass GO. More seriously if the problem is repeatable, as most bugs are, then rerun the program with Discipline on, and if you have a computer with an 020/030 CPU then turn on **"Trace Jumps"** in the **Go"** menu. If you are not me, then avoid trying to step into the ROM code as you will just get lost in the Rats nest that it has become. Look at the code in your program, read the accumulated tech notes, Inside Macintosh Volumes 1 through 5, etc until you can pin the blame upon your program. All else failing, take a coffee or a soda break or get some fresh air.

Terminating the current debugging session

When the sushi has hit the fan or the magic Bomb box has appeared, you should consider the "Boot System" or "Flush System and Boot" commands as viable alternatives to continuing the current masochism. If you don't want the Volumes flushed on a Mac II/IIx, ..., then use the Reset switch on the side of the Machine.

Terminating an Application (Multifinder mode only)

To do this the PC must be within your APPLications heap zone, so adjust it first if necessary, then select the Shutdown APPL command from the first menu.

A last Word - don't say I didn't tell ya so

Make a copy of your System file on your hard disk, and keep a Bootable floppy disk with some recovery utilities handy. MPW's *Rezdet* will detect corrupted System files, etc.

A Quick Reference for The Debugger V2 - 1/4/93 © Steve Jasik 1993

Installation Procedure

Debugger Tech Note #0 is the guide to installing The Debugger, read it.

Multiple Monitor Operation

Mac II, etc - Debugger takes over the startup Screen. Option-Monitors in Control Panels to set.
In Monitors, drag the little Mac Icon to the Monitor that you want *The Debugger* to appear on.

RunDbgr versus XDbgr_Startup

Support for RunDbgr is being phased out as it has no use in System 7.
You need at least 4 Meg of RAM to run *The Debugger* in the background and 8 Meg or more is preferred to get any real work done. You can **NOT** Launch *The Debugger* after MultiFinder.
You may change the name of xDbgr_Startup so that it loads earlier in the INIT load sequence.
xDbgr_Startup "launchs" a file by the name of "*The Debugger* ", so do not change it.

Macsbug, INITs and Other Gotchas

The *Macsbug* in "put files in System folder" installs itself in high memory and records patches to the Traps until the Finder is launched. It uses 8K of memory. To see who patched the Traps, check out the "ROM Patch Info" command in the Display menu.
You may install Apple's Macsbug as Macsbug, and rename my *Macsbug* to "Disassembler".
When *The Debugger* starts up it makes a copy of the "low memory globals" (trap vectors, etc.) so that it **will not be affected by patches to the traps** that are set by programs that come after it.

The Debugger is not compatible with all INITs. See Debugger Tech Note 0 for details.
To avoid interference between the command-option key equivalents in QuicKeys™ and those in *The Debugger*, you may wish to rename QuicKeys™ to zQuicKeys™ .

Menus and Commands, General

Command-option key equivalents show up as **Omega**-Key (Ω) in the Menus.
When the key equivalent is a **lower case letter**, Shift-Command-letter will result in a different function (Cmd-B is a simple Bkpt, Shift-Cmd-B is a Bkpt with the current action clause attached to it.)

Some unshifted/shifted menu items options are presented in the menu in the format "unshifted/SHIFTED command." For example, in the *Tables* menu, the first menu item is **Record/ALL names**, which lists all the known type names if the shift key is held down, and only the built in Record names if is not.

Menu items that are check items show up in **Bold face**; a check mark indicates it is selected.

Many commands check for a selection in the front window, and will use it if present.

Pressing the 'esc' key will clear any selection in the front window.

If nothing is selected, then the command will bring up a simple type in dialog box just below the menu bar. You can **paste into these dialogs** with **Cmd-V**; and dismiss them with a Cmd-period.
Clicking in the **"Cmd:"** box is equivalent to a return.

Use **Cmd-period** to abort a command or stop continuous Step mode, etc.
The first item in the menu bar is the amount of free space in *The Debugger*'s Heap zone.
The first menu contains a **Help** command.

The Find command - Searching for Text in a Window

If some object is selected in the front window, then the Find command (Cmd-F) accepts it as the search pattern and looks for the next occurrence of it as a token (separated by delimiters).

If nothing is hilited in the front window, then it brings up the find dialog window .

An alternate way to a search for a pattern (case insensitive, no delimiters) is to copy a string (Cmd-C) and then **Cmd-G**.

Windows & Tasks, General

All windows (not dialogs) use a custom text editor that supports the display of large files, tabs and Undo/Redo. It's behavior is similar to that of the MPW Shell editor.
Cmd-delete deletes to the end of file. Triple clicking will hilight a line, and double clicking on a left paren ('(', '{', '[') or quote mark, etc will hilight to the matching right paren, etc.
Use **Option-Click** to obtain a popup menu of the windows to bring one to the front.
In *The Debugger* most of the windows are read only. You can copy from them.

A quick way of transferring a selection to the "-Notes-" window is to use **Cmd-N**.

Windows that are **task relative** are prefixed with "∂taskname."
A **task** is a process (APPL or executable resource). The list of tasks known to *The Debugger* is displayed in the **"-Task & File Info-"** window (Ω-T) along with the list of open files.
The last item in the Menu bar is the number and name of the task you are in. It will change as you switch from one task relative window to another. Commands that are task relative (such as display Code | Data block) will search tables, etc. relative to that task.

To debug a task **symbolically** you must prepare an **auxiliary file** that contains symbol information that resides in the **same folder from which the task will be executed**.
The aux file may be: a Nosy "snt" file, an MPW or MDS style ".map" file, an MPW V3.x ".SYM" file or the Think C PROJ file itself (see page 32 of *The Debugger* manual).

Debugger, Entry/Exit

Cmd-E to exit *The Debugger* .
To enter *The Debugger*, press **Option-** or the Programmers Switch, or press Cmd-PowerKey if you are using the Programmer's Key INIT.

When in *The Debugger*, use the ~ (tilde) key to view the PP's screen (swap screen mode).

On Entry to The Debugger from the PP

On a context switch into it, *The Debugger* puts one or more lines into the **"-Notes-"** window to show why you entered it, updates the **"-Registers-"** and **"-Stack State-"** windows, and a window of the procedure containing a disassembly (or source) of the current PC.
The menubar task indicator is set to the task of the procedure containing the PC.
To reset the display to the **current PC**, hilite the value of the PC in **-Registers-** and Cmd-D.

Source Level Debugging (MPW V3.x .SYM file or Think C™ project)

For a Think C project, the **source files must be on the same volume** as the "project" file, while with MPW ".SYM" files, the path to the source is specified in a ".dsi" file (DTN #3).

Use the "Source only windows" command (Ω-V) to set the default mode of display.
The first line of the window gives the name of the file containing the source of the procedure.

Use a **Command-click** (mouse down) to toggle a window between Source only and source with interspersed assembly. In the source only windows, a"Σ" in column 1 denotes the beginning of a statement. In LightSpeedC **Compile with "Use Debugger" on** and run with it OFF.

Displaying procedures

Hilite the name of a proc and **Cmd-D**, or Cmd-D with nothing hilited and you will get a type-in dialog box. Typing in the leading characters of a procedure name will result in a window with a list of names that match, or one with code of the procedure if the leading chars only matched on one name.
More often in *The Debugger*, you want to look at an active proc, i.e. one that is in the call chain.
Activate the **"-Stack State-"** window, find the proc you want, hilite the "=nnn" and Cmd-D. Not only is this slightly faster, but it also switches *The Debugger* to the task the proc is in and positions the selection to the point of the call.

Breakpoints

To set or clear a breakpoint, hilite a line in a procedure window and **Cmd-B**.
To attach the current "action clause" to a breakpoint, **Shift-Cmd-B**.

To define an "action clause", hilite it and **Ω-B**. It will become the current "action clause".

To change the current "action clause", de-hilite any selection in the front window and **Ω-B**.
Then enter the number of the "action clause" you wish to make current.

To delete an "action clause", de-hilite any selection in the front window and **Shift-Ω-B**.
Then enter the number of the "action clause" you wish to delete.

Do not set a breakpoint in ROM patch code that The Debugger may execute!

Debugger Cheat Sheet

In Application: *Rev 6/11/95*

Option- **Break into Debugger (after _WNE/_GNE)**

In Debugger: **(if a code window is frontmost, then ⌘ key is optional)**

Ω	*option and ⌘ keys both held down*	*Macsbug Equiv*
⌘-. or ESC	Abort/cancel current command, dialog, etc	
~	Toggle to view user's screen	~
⌘-D	Display (hilited) Procedure	il
⌘-W	Display (hilited) variable value (global or local)	
⌘-Space	Display data value of memory (type_@nnnn, ...)	dm
Shift-⌘-U	Display Local vars Window (all active procs)	
Ω-U	Display Local vars Window (current proc)	

click on Pencil Box in Window Title to enable type-over of data values

⌘-8	Toggle window for dynamic update (⌘-space & Values windows)	
⌘-B	Set/clear breakpoint on hilited line	br
Ω-B	Define or remove a Breakpoint "action clause"	
Shift-⌘-B	Set breakpoint with current action clause attached	
Ω-W	Set/clear Watchpoint (memory change)	ss
Ω-P	Go (execute) until hilited line is reached	
⌘-E	Exit to program	g
⌘-,	Step over current line of code	so
⌘-?	Step into current line of code	s
⌘-;	Step continuous (⌘ key down to pause, ⌘-period to stop)	
⌘-H	Display Current Heap Zone (shift-⌘-H for System heap)	hd
Ω-G	Find the heap zone and block containing a given address	wh
Ω-T	display list of Tasks, open Rsrc files, Open files	
⌘-[	Expression Calculator - evaluate the value of an 'expression'	
Shift-⌘-[	Execute "block of statements" or a procedure call or *DCMD*	
⌘-minus	Convert Hilited Hex number to decimal	
Ω-F	Execute hilited ".dsi" file command	
⌘-=	Unstructured dump of 68K code	il
Shift-⌘-=	Unstructured dump of PowerPC code	
Ω-2	Side Door to Shell (MPW or ObjectMaster™)	

In a Source code window:

To Set/clear breakpoint move the cursor to the left edge and it will change to a STOP sign with a bullet in the center. Clicking the mouse will toggle a Breakpoint.

Holding down the ⌘ key and clicking the mouse button will toggle between a source code only and source with assembly interspersed.

Anywhere:

To get a Popup Menu of the list of windows, hold down the Option key and press the Mouse button.

To Toggle a Debugger Flag:

Double-click on the flag Name in the '-Dbgr Status-' window.

To reset the hilited line to the current value of the PC (Program counter):

Double-click on the *VALUE of the PC* in the '-Registers-' window and ⌘-D

in MPW Shell (IBS commands):

Opt-⌘-Z	Hint hilited procedure
Opt-⌘-B	Patch Link
Opt-⌘-P	Show IBS project Status

Cmd-Option-Click processing

In a Source code window:

Hilite a name and Ω-click will produce a Popup Menu with the choices

Display as a Procedure (⌘-D) or as a Variable (⌘-W)

In all other windows Ω-clicking on an address or "type@address"

OR in the title bar of a "type@address" window

will produce a 'View as' Popup Menu which will let you cast the address to the type you want.

Types below the default type may be set via the '.dsi' command:

=X define types in 'View As' - one per line or comma seperated list

RoutineDescriptor

ParamBlockRec

Fixed, Cstring

....